Student Comments from the October 2008 SPICE Training Class

During the SPICE Training Class of October 2008, held in Pasadena, a "Peer Feedback" form was passed out in an attempt to elicit comments on how to improve such training classes, how to improve the SPICE system, and how to improve the NAIF Group's operations. The form contained a list of possible areas for improvement, and some students commented on some of these specific items. In addition, two students provided more general feedback. Here we first summarize the specific comments and then the general comments. There is no particular order to the summaries, and you will see that some topics received comments from two or more students. We've provided the comments pretty close to the original handwritten student notes. Our apologies if we missed or misinterpreted any comments.

In some cases we don't quite understand the comment made, but we try to reproduce all of them here anyway.

A few NAIF notes and comments are provided within square brackets [].

Specific Comments

Metadata that are often (sometimes) provided with kernels should include a URL pointing to where the file came from and can be retrieved.

Provide a tutorial outlining how one can (may? must?) deploy one's own APIs (subroutines), in particular, how and where to deliver headers. [NAIF did not understand this comment.]

Provide some sort of Source Forge equivalent where SPICE users could contribute their own code for use by others.

Provide more (more difficult) components to the hands-on programming lessons: they were generally too easy for people already somewhat familiar with SPICE.

The class was very useful for getting a complete overview of the SPICE system.

Provide a FORTRAN 95 version of the SPICE Toolkit.

Provide a perl interface to the SPICE Toolkit.

Examples contained in the code documentation (the so-called "headers") are very helpful and well done.

It was very nice all documentation is available on-line, rather than killing trees. A great documentation collection.

Would like to have wireless Internet connectivity to use during class breaks.

The “hands-on” programming lessons were well thought out and nicely dispersed throughout the class; well geared to those trying to learn how and when to use the various functions.

Great class; I’m glad to have had the chance to take the class without first having an extensive background in it.

Would like to see “GoView” or “Gravity 3D” developed (as a space geometry visualization program).

Make the “chronos” time conversion program available through the web using REST (Remote Software Toolkit).

Add perl and python interfaces to SPICE. Much of the string handling capability provided in the Toolkit can be done better in perl.

Thanks for including a lesson on where/how to find various SPICE components.

Provide more difficult programming examples (for more advanced students).

Offer an advanced class using “real life” operations scenarios.

Need a better naming convention for functions that supersede old functions. For instance, “subslr” has superseded “subsol,” but there is no easy way to find this out. As is one has to check the documentation of both APIs to figure this out.
[Instructor Jorge Diaz suggests adding a link to the new routine where, in the deprecated one, we say it is deprecated.]
[Perhaps NAIF should produce a stand alone document that lists all deprecated routines and their replacements.]

I like the headers.

Provide a document with SPICE spacecraft IDs and showing all the spacecraft that used that ID at one time or another.
[This is the conundrum resulting from the fact that the NASA control authority for spacecraft numbers reuses IDs, saying they never were meant to be “forever” assignments.]

In the “Icy Required Reading” you need to show how to load kernels in order to run the examples.

Regarding the class:
- Would have liked wireless Internet access
- Don’t stand in front of the screen (when presenting tutorials)
- Like having the programming lessons intermixed with the tutorial presentations
- Leave time for a break each morning and afternoon

Not sure if a C compiler or CSPICE is needed to use “ICY.” Need to better explain this.

Of the future capabilities mentioned, like the Frame [Creation and] Visualization Tool and completing the integration of a star catalog capability [has been sitting at about 90-95% done for many years]

Simply the best training class I have ever attended (best prepared, organized, complete).

Like the idea of a SPICE Source Forge location, to contribute and obtain user-made capabilities.

Very good documentation: complete, navigable and with good examples. But would like to see an improvement to the permuted index search capability.

If you’re going to build a GUI, build one for the event finder subsystem [a.k.a. geometry finder] currently under development.

Aside from the hands-on lessons, the biggest advantage of the class was the in-person interactions with the software developers. This is especially true when one tries to understand some nuances such as the difference between surface intercept point versus nearest point.

A Source Forge type of arrangement would be helpful, especially if organized by type of routine and by programming language.

Looking forward to the new shape modeling software now under development.

As a visual person, it would be great to have more tools to visualize SPICE data, but not necessarily GUI interfaces for the toolkit. It seems like all the information is there to fully model the solar system, all the spacecraft and communications systems at any given time, or over a specified range of times.

I do have one question left from the class. Many of the tutorials and lessons were designed to calculate for a unique epoch. What about continuous time intervals? I think I saw some mention of looping over time steps. Is that the only way? [The IDL and MATLAB Toolkits (“Icy” and “Mice”) offer vectorized computations for most commonly used computations. There is no equivalent offered in the FORTRAN and C Toolkits.]

<u>General Comments</u>

Would have been interesting to get more background on how SPICE was/is developed; things like “we have had ten developers over time, there are X million lines of code,” etc. It would really be interesting to understand how SPICE is developed day-to-day… things like how [development] priorities get set, how changes are proposed, how bugs are tracked, who is responsible for what areas, and how things get tested. Even if this were just 15 minutes worth.

Would also like to have heard how various people use SPICE on a day-to-day basis to make their jobs easier. Also would like to know he kinds of tools and processes they have in place to do things like updating or downloading new SPICE kernels.

You all have done a great job producing the data and the tools to use it. I am very impressed. Thanks for spending the time to help us learn how to use it, and for being so open to suggestions. It was a good introduction to SPICE. After I have used it a bit more, and feel more confident working with the toolkit, an advanced class would be of interest. Keep me posted.